# Functions and Fractals: Sierpinski triangles

1	
111	
111	
111_111	
11	
111111	
1111 111 111 111 111	
1 1	
111 111	
1 1 1	
111_111111_111	
11111	
1	
111111	
1111111	
111_111111_111	
11111_1_1_1_1_1_1_1	
111 111 111 111 111 111 111 111	
1 1 1	
111111111111	
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	
111_111_111_111_111_111_111_111_111_111_111_111_111_111_111_111	

## Sierpinski Triangle

The Sierpinski Triangle is a pretty fractal which consistes of layers of self-similar triangles, nested inside each other. This challenge involves the construction of such triangles, in the form of ASCII Art. The restriction is, that you need to accomplish this with functional programming, and you cannot declare even local variables!

We have to deal with real world constraints, so we cannot keep repeating the pattern infinitely. So, we will provide you a number of iterations, and you need to generate the ASCII version of the Sierpinski Triangle for those many iterations (or, levels of recursion). A few samples are provided below.

### Iteration #0

In the beginning, we simply print a triangle which points upwards. There are 32 rows and 63 columns in this matrix. The triangle is composed of underscores and ones as shown below.

1	
111	
11111	

1111111
111111111
1111111111
11111111111
1111111111111
11111111111111
1111111111111111
111111111111111111111111111111111111111
111111111111111111111111111111111111111
111111111111111111111111111111111111111
111111111111111111111111111111111111111
111111111111111111111111111111111111111
111111111111111111111111111111111111111
111111111111111111111111111111111111111
11111111111111111111111111111111111
11111111111111111111111111111111111
111111111111111111111111111111111111111
111111111111111111111111111111111111111
111111111111111111111111111111111111111
111111111111111111111111111111111111111
111111111111111111111111111111111111111
111111111111111111111111111111111111111
111111111111111111111111111111111111111
111111111111111111111111111111111111111
11111111111111111111111111111111111
111111111111111111111111111111111111111

# Iteration #1

The "Fractalization" now begins. We create a new triangle, which points downwards, and its vertices coincide with the midpoints of the outer, upward-pointing triangle. The ones are flipped into underscores. Note, that the original upward-pointing triangle has now been split into four segments: one downwardpointing triangle, filled with underscores - and three triangles which point upwards and are filled with ones.

	1		
11111			
	1111111111		
11111111111			
11111111111111			
111111111111111111111111111111111111111			
111111111111111111111111111111111111111			
11111111111	1111111111111 11111111111111		
111111111111	 11111111111111		
1111111111111	111111111111111		
111111111111111	111111111111111		
11	11		
111	111		
11111	11111		
1111111	1111111		
111111111	111111111		
1111111111	1111111111		
111111111111	111111111111		
11111111111111	11111111111111		
11111111111111111	11111111111111111		
1111111111111111111	1111111111111111111		
1111111111111111111111	11111111111111111111111		
111111111111111111111111	1111111111111111111111111		
1111111111111111111111111	11111111111111111111111111		

### Iteration #2

We repeat the process on the three smaller upward-pointing triangles created at the end of Iteration #1. We create a downward pointing triangle inside each of those.

		1		
	111			
11111		111		
	111:	1111		
	11111:	111111		
	11111111111			
	1	1		
	111	111		
	11111	11111		
	1111111	1111111		
	111111111	111111111		
	11111111111	11111111111		
	1111111111111	1111111111111		
	111111111111111	111111111111111		
	1		1	
111		1:	11	
		11:	111	
1111			1111	
11111	11111	1111:	11111	
		1111111111		
111111	1111111	111111	1111111	
1111111	 1111111111111111		1111111111111	
1	1	1	1	
111	111	111	111	
11111	11111	11111	11111	
1111111	1111111	1111111	1111111	
111111111	111111111	111111111	111111111	
1111111111	11111111111	11111111111	11111111111	
111111111111	1111111111111	1111111111111	1111111111111	
111111111111111				

### **Input Format**

One Integer N which is the Iteration Number for which you need to generate the Sierpinski triangle, in accordance with the triangles displayed above.

Generate the N<sup>th</sup> triangle in the series shown above.

# **Input Constraint**

N <= 5

# **Notes about the Triangle**

As in the figures above, the canvas has a total of 32 rows and 63 columns. The outermost, upward-pointing triangle has a perpendicular height of 32 characters. The height of each of the downwards-pointing triangle, drawn in each iteration, is half of the upward-pointing one in which it is drawn.

### **Output Format**

The N<sup>th</sup> triangle of the series shown above. The output will consist of 32 rows and 63 columns, and will be composed of ones (1) and underscores as in the triangles above.